**Introduction:**

The task isn’t coding task, it’s more of DevOps/System integration task.

The project provides 3 routes (URLs):

* <http://localhost/>
* <http://localhost/sleep/>
* [http://localhost/sleep/<#seconds](http://localhost/sleep/%3c#seconds)> seconds can be set with any integer value you would like.

The interesting route for your task is the second/third route – it is doing some work and then return an output to your browser.

After you done testing the routes lets dive to the task.

Let’s think about a system that cannot have downtime for instance “Moodle” system at your college, you are now managing its releases deployment. Moodle release manager told you, that they developed new feature and you must deploy it, but you can’t interfere the users that are using the system.

**The problem:**

You must deploy new code, but you can’t have downtime, which means:

On one hand if a user is working on the system he shouldn’t feel any change. On the other hand, new logged in users should already have the new features.

**The task:**

Your task is to “deploy” the containers (represent Moodle system) with no downtime. As already described before, users shouldn’t feel any inconvenience.

The project is made of two services:

“nginx” – is the gateway server which handles http requests.

“web” - provides “UI” and “business logic” for this project (based on flask application) – you can read about flask a bit.

How can you test yourself?

Create “get” request for the “sleep” url with 60 seconds ( <http://localhost/sleep/60> ), once you see that the “website” is loading the page (sleeping) deploy new release (To make it more interesting you can change the code and make another route in app.py) and see if the new code is running. To validate “zero downtime deployment” you should still get the message “I was sleeping for 60 seconds” in the first window.

Please describe each step you make and explain how your system works in “docx” file.

Please use your Github repository with meaningful commit messages (if needed).

You may use any “Open source” package/module.

**Docker world:**

For this project you will need docker-compose (depends on docker).

Docker compose gives you the ability to bring up multiple docker services.

You can find more info on official docker site: <https://docs.docker.com/compose/install/>

**Useful docker commands:**

For your knowledge, any command had “—help” flag you can use and read.

* Build images: docker-compose build
* Run containers: docker-compose up
* Run containers in background: docker-compose up -d
* Stop containers: docker-compose stop
* List running containers: docker-compose ps